BIGhybrid - A Toolkit for Simulating MapReduce on Hybrid Infrastructures

Julio C. S. dos Anjos
Federal University of
Rio Grande do Sul, Brazil
Institute of Informatics - PPGC
CNRS - INRIA/LIP, ENS Lyon - UCBL
Email: jcsanjos@inf.ufrgs.br

Gilles Fedak
INRIA/LIP, Univ. Lyon, France
Email: Gilles.Fedak@inria.fr

Claudio F. R. Geyer
Federal University of
Rio Grande do Sul, Brazil
Institute of Informatics - PPGC
Email: geyer@inf.ufrgs.br

***Abstract*—Cloud computing has increasingly been used as a platform for running large business and data processing applications. Although clouds have become highly popular, when it comes to data processing, the cost of usage is not negligible. Conversely, Desktop Grids, have been used by a plethora of projects, taking advantage of the high number of resources provided for free by volunteers. Merging cloud computing and desktop grids into hybrid infrastructure can provide a feasible low-cost solution for big data analysis. Although frameworks like MapReduce have been conceived to exploit commodity hardware, their use on hybrid infrastructure poses some challenges due to large resource heterogeneity and high churn rate. This study introduces BIGhybrid a toolkit to simulate MapReduce on hybrid environments. The main goal is to provide a framework for developers and system designers to address the issues of hybrid MapReduce. In this paper, we describe the framework which simulates the assembly of two existing middleware: BitDew-MapReduce for Desktop Grids and Hadoop-BlobSeer for Cloud Computing. Experimental results included in this work demonstrate the feasibility of our approach.**

I. Introduction

Mankind has been producing over increasing amount of data. According to IDC[1], by 2020 there will be around 40 Zettabytes (40,000,000 Petabytes) of data that will require processing of some sort. This data volume requires processing capabilities beyond those that current IT infrastructure can provide.

MapReduce (MR) [1], a programming framework proposed by Google and currently utilized by many large companies, has been employed as a successful means for data processing and analysis. Hadoop, the most popular open-source implementation of MR [2], abstracts task parallelism management from programmers who only need to implement their applications as *Map* and *Reduce* functions. Cloud computing has increasingly been used as a platform for business applications and data processing [3]. Cloud providers offer Virtual Machines (VMs), storage, communication, and queue services to customers for which they pay an hourly fee. These resources can be used for deploying Hadoop clusters for data processing and analysis.

When it comes to data processing, the computing power offered by other types of infrastructure is also of interest. Desktop Grids (DG) [4], for instance have numerous users around the world who donate idle computing power to multiple projects. DG have been applied in several domains such as biomedicine, weather forecasting, and natural disaster prediction. Merging DG and Cloud Computing (Cloud) into Hybrid Infrastructures could provide a more affordable mean for data processing. However, although MR has been designed to exploit commodity hardware, its use on hybrid infrastructure poses some challenges due to large resource heterogeneity and high churn rate typical of such environments.

The adaptation of existing MR framework or the development of new software raises research issues related to *a priori* data splitting and distribution, strategies to avoid the communication between the infrastructures, tasks and data co-scheduling, fault and sabotage tolerance, data privacy, among other issues and more. Moreover, using real testbeds to evaluate the use of MR on hybrid infrastructure presents considerable challenge due to the lack of reproducibility of the experimental conditions in DG and the complexity of fine-tuning Cloud software stacks.

This study introduces BIGhybrid, a toolkit for simulating MR on hybrid environments, with a focus on Cloud and DG. We analyze the characteristics of hybrid MR runtime environment and design the simulator accordingly. The simulator is based on SimGrid [5] and leverages solutions proposed in the scope of the *MapReduce* ARN project [6].

It is able to simulate two middleware for two distinct infrastructures: BitDew-MR [7], [8] for Desktop Grid Computing and Hadoop-Blobseer [9] for Cloud computing. BIGhybrid has several desirable features: it is built atop of MRSG, a validated Hadoop simulator [10], and MRA++, a simulator for heterogeneous environments [11]; it has a trace toolkit that enable analyze, monitor and graphically plot the task executions; it is a trace-base simulator able to process real infrastructure availability traces [12]; and its modular design allows for further extension. BIGhybrid can be used for evaluating scheduling strategies for MR applications on hybrid infrastructures. We believe that such a tool is important to researchers and practitioners working on big data applications and scheduling.

The rest of this work is structured as follows. Section II provides an overview on the MR framework and other systems used. In Section III, we describe the characteristics of hybrid infrastructures. Section IV introduces BIGhybrid and Section

[1]IDC's Digital Universe Study, sponsored by EMC, December 2012.

V describes the evaluation methodology. The conclusion and suggestions for future work are summarized in Section VI.

II. Background

This section presents MR and introduces the implementations of both Hadoop-BlobSeer and BitDew-MapReduce, that will be used on BIGhybrid simulator.

A. MapReduce

MR is a programming framework that abstracts the complexity of parallel applications by partitioning and scattering data sets across hundreds or thousands of machines, and by moving computing closer to data [2]. Figure 1, adapted from [2], shows the MR data flow. The *Map* and *Reduce* phases are handled by the programmer, whereas the *Shuffle* is created while the job is being carried out. The input data is split into smaller pieces called *chunks*, that normally have a size of 64 MB. The data is serialized and distributed across machines that compose the Distributed File System (DFS).

When running an application, the master assigns tasks to workers that then run each processing stage. The machine that receives a *Map* task, handles a *Map* function and emits *key/value* pairs as intermediate results that are temporarily stored in the workers' disks. The execution model creates a computational barrier, which allows tasks to be synchronized between the producers and consumers. A *Reduce* task does not start its processing until all the *Map* tasks have been completed. A hash function is applied over the intermediate data to determine which key partitions will compose a *Reduce* task. Each key partition is transferred to one machine, in prefetching mode, during the *Shuffle* phase, to execute the next phase. After a *Reduce* function has been applied to the data, a new resulting *key/value* pair is issued. Then results are stored in the distributed file system and made available to the users.

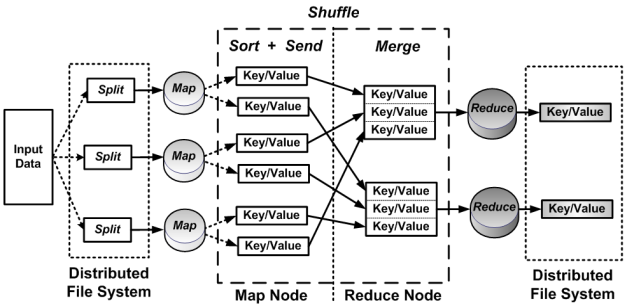

Fig. 1. MapReduce data flowchart model

The MR Hadoop provides management mechanisms, data replication and execution control. The MR has a management architecture based on the master/worker model, while a slave-to-slave data exchange requires a P2P model [2]. When the task runtime on a machine is greater than average for the cluster, the machine running the task is characterized as a straggler. If a machine is characterized as a straggler after the first task distribution, it will not receive new tasks to processing in free slots.

B. Hadoop-BlobSeer

BlobSeer is a DFS that manages a huge amount of data in a flat sequence of bytes called BLOBs (**B**inary **L**arge **O**bject**s**). The data structure format allows a fine-grained access control. BlobSeer emits file version control which enables incremental updates on data files, and high throughput with concurrent reads, writes and updates data. The data *chunks* have a fixed size of 64 MB so that they can maintain compatibility with the structure of the Hadoop file system.

The DFS on Hadoop (HDFS) was replaced in its integrally by BlobSeer. An API implements the calls of Hadoop for the BlobSeer File System (BSFS). The name space manager is centralized, and it keeps the name space of BSFS for mapping files for BLOBs. However, this data structure is completely transparent for the Hadoop system and its MR users. The classical execution of MR on Hadoop was not changed and explores data locality similar to HDFS. The BSFS provides a data replication using a flat structure. The blocks are distributed for local storage in machines with a load-balancing strategy. The fault-tolerance mechanism is a simple data replication across the machines. However, it does not explore replication across racks as in HDFS.

C. BitDew-MapReduce

BitDew [7] is a middleware that exploits protocols like P2P, http, BitTorrent and ftp, and selects the best protocol depending on data size. BitDew architecture is decentralized and has independent services for *Data Scheduler*, *Data Catalog*, *Data Repository* and *Data Transfer*. These services are accessed via three API: *Active Data* to control the behavior of the data system, such as replication, fault-tolerance, data placement, lifetime, protocols and event-driven programming facilities; *Transfer Manager* to manage the concurrent file transfer completions and the concurrency level of transfers; *BitDew* to provide functions that create slots and carry out data management.

The *Data Catalog* maintains a centralized and updated meta-data list for the whole system in a *Distributed Hash Table*. The model includes both stable and volatile storage. The stable storage is provided by stable machines or Cloud Storage like Dropbox and Google Drive, and the volatile storage are local disks of volatile nodes. The MR implementation [13] is an API that controls the master and worker daemon programs. This MR API can handle the *Map* and *Reduce* functions through BitDew services. The data locality from Hadoop MR was implemented like a data attribute to support the separation of the input data distribution from the execution process.

Result checking is controlled through a major voting mechanism [8]. In the Hadoop implementation when the network experiences unavailability, a heartbeat mechanism signals to the master that the host is dead. However, in BitDew the network can be temporarily offline without undergoing any failure. The fault tolerance system needs a synchronization schema, as pointed out by [14] where transient and permanent failures can be handled. A barrier-free computation is implemented in BitDew to mitigate the host churn behavior [13]. In BitDew (unlike the case of the classical MR implementation), the computation of *Reduce* nodes starts as soon as intermediate results are available.

III. HYBRID INFRASTRUCTURE

Table I, summarizes the main architectural features of Cloud-BlobSeer, BitDew-MapReduce and the Hybrid MR environment. Hybrid infrastructure enables the use of highly heterogeneous machines, and stable and volatile storage to avoid data lost. The set of data-distribution strategies applicable to a given scenario depends on how much bandwidth is available. Two independent DFS implementations are required to handle data distribution under two scenarios, namely low-bandwidth and high-bandwidth. Hybrid infrastructure uses an orchestrator to manage results and data input from users. It can be decentralized to improve data distribution over network. In special case of Cloud and DG, fault tolerance mechanisms have different policies to detect faults. A more specialized system is applied on DG due to node volatility.

Figure 2 illustrates the solution proposed to model a hybrid system and introduces *Global Dispatcher* and *Global Aggregator* that will be used on BIG*hybrid* simulator. The *Global Dispatcher* located outside the DG and the Cloud has middleware functions for receiving both job tasks and input data from users. It is a centralized data storage system that manages policies for split data and distribution, in accordance with the needs of each system. The working principle is similar to the publish/subscribe service where the system gets data and publishes the computing results. This approach is simple, but creates a potential network bottleneck. BIGhybrid simulator will enable the study of several strategies to determine the best data distribution and resource allocation on MR applications in hybrid infrastructures.

Global Aggregator receives all *key/values* of *Reduce*, and keys with the same index in each system are joined to last *Reduce* function in order to create a consistent result. The BIGhybrid simulator will help to choose the best strategies to achieve this goal.

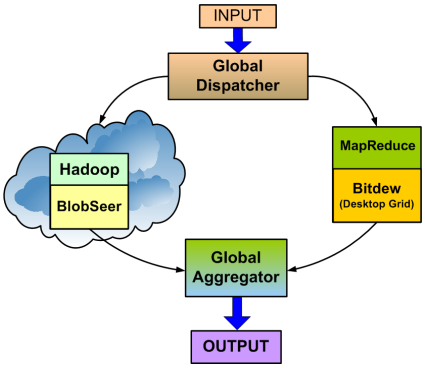

Fig. 2. Hybrid Infrastructure

Iterative MR computations, necessary on *Global Aggregator*, are not supported by an original MR model. It is not an easy task to combine all *Reduces* from heterogeneous platforms, although it is possible to make a new stage for MR [15]. A possible approach is to use the *MapIterativeReduce* [16] which creates an *Aggregator* to collect all the outputs of the *Reduce* tasks and combines them into a single result. At the end of each iteration, the reducer checks if it is the last or not. However, according to [17], this schema might be inefficient for large workloads. BIGhybrid enables the study of variations and patterns to be implemented for aggregation module.

IV. BIGHYBRID SIMULATOR

The idea behind BIGhybrid simulator is to optimize MR applications in order to provide a cloud service with the available resources of a DG system. BIGhybrid is modular and it is built on top of Simgrid [5], which is a simulation-based framework for evaluating cluster, clouds, grid and P2P (peer-to-peer) algorithms and heuristics. Different from others simulators, BIGhybrid has two independent systems, through which it is possible to use different configurations for DFS, schedulers, input/output data size, number of workers, homogeneous and heterogeneous environments, combine two different platforms, and make the simulation parallel. BIGhydrid generates traces from each system to allow individual or collective analyzes in the same time line.

BIGhybrid is built on two components described in previous work: MRSG (*MapReduce* over SimGrid) that simulates Cloud-BlobSeer with Hadoop; and MRA++ (MapReduce Adapted Algorithms to Heterogeneous Environments) that simulates BitDew-MapReduce. Figure 3 illustrates the architecture of BIGhybrid, which comprises four main components: input data management (Global Dispather), Cloud-BlobSeer module, BitDew-MapReduce module and integration module for results (Global Aggregator). More details about MRSG simulator and MRA++ can be found in [10] and [11].

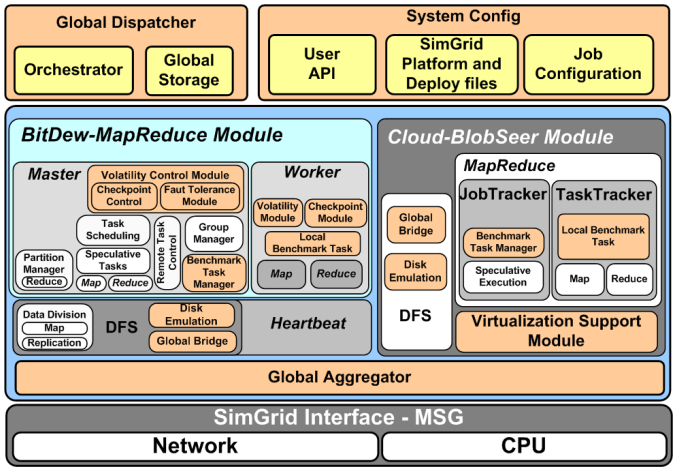

Fig. 3. BIGhybrid Simulator Architecture

A user can specify an input function for each system and for individual *Map* and *Reduce* functions. In the next release version it will be possible to build platforms for real infrastructures using Failure Trace Archive [12]. This means, the BIGhybrid enables 256 settings of configurations in the same simulator. It is possible to make adjustments to several kinds of strategies and configurations in both Cloud-BlobSeer with Hadoop and BitDew-MapReduce, to find a load balance without data loss and with suitable strategies to achieve an efficient data partition between the two environments.

A. Cloud-BlobSeer Simulation Module

The Cloud-BlobSeer simulation module reproduces the behavior of the MR platform, and invokes SimGrid operations whenever a network transfer or processing task must

TABLE I. COMPARISON AMONG MR SYSTEMS

Characteristics	Cloud-BlobSeer with Hadoop	BitDew-MapReduce	Hybrid-MapReduce
Heterogeneity	Moderately	High	High
Network	High Bandwidth	Low Bandwidth, distributed cache	Hybrid Bandwidth
Local Storage	Distributed	Remote (Cloud Storage) + local	Local and distributed + Remote Cloud Storage
Management	Master/Slave	Master/Slave	Hierarchical Orchestrator
Application Profile	Any	Low Communication on *Shuffle* phase	Optimized for all size archives
File System API	Posix	Tuple Space model	Hybrid (Posix + Tuple Sace)
Hosts	Stable	Stable and Volatile	Stable and Volatile
FT mechanism	Data and Task Replication	Data Replication and transient failure support	Data and Task Replication, and transient failure support
Computation	Hadoop Compatilble	Barrier-free	Hybrid

be performed, without modifying SimGrid source code. This simulation follows the Hadoop implementation, which a heartbeat mechanism to control job execution. This architecture is the following: API of input users code, DFS, *MapReduce* functions, master (Jobtracker) and slaves (Tasktracker).

The DFS is implemented as a matrix that maps *chunks* to nodes. The master node knows where each *chunk* is placed, exactly as it happens in the real implementation. Moreover, each *chunk* can be linked to more than one node, which allows to simulate *chunk* replicas. SimGrid is responsible for the simulation of all network communication and task processing in our implementation. Cloud-BlobSeer simulation only implements the node distributions in one rack. The next version of BIGhybrid will use the storage simulation API of SimGrid, on *Disk Emulation Module* to simulate the storage behavior. As of writing disk simulation is specified as an I/O cost in the configuration file in User API.

The virtualization of machines behavior is not simulated in current version of BIGhybrid, but is abstracted as an additional task cost. The virtualization support module will be integrated in the future by the SimGrid virtualization support, as described in [18], and migrating virtual machines, as described in [19].

B. BitDew-MapReduce Simulation Module

BitDew [7] is a middleware for large scale data management on hybid distributed computing infrastructures. Its runtime environment supports the use of multiple file transfer protocols: either client/server (http, ftp, scp), P2P (bittorrent), Grid (through the SAGA API) or Cloud (Amazon S3, Dropbox). A set of BitDew services handle many high level data management issues: data index and meta-data catalog, fault tolerance, reliable file transfer, data scheduling, replication, data life-cycle, collective communication, data collection management, event-based programing model and more.

The implementation of MapReduce over BitDew targets firstly Desktop Grid systems [13], proposing mechanisms to alleviate host churn, unavailability of direct communication between the hosts and lack of host trust. The implementation relies on master and worker daemon programs. This MR API can handle the *Map* and *Reduce* functions through BitDew services. The data locality from Hadoop MR was implemented like a data attribute to support the separation of the input data distribution from the execution process.

A function controls result checking through the major voting mechanism as in Moca [8]. When the network experiences unavailability, a heartbeat interval signals to the master that the host is dead, in the Hadoop implementation. However, in BitDew the network can be temporarily offline without undergoing any failure. The FT needs a synchronization schema, as pointed out by [14] where transient and permanent failures can be handled. A barrier-free computation is implemented in BitDew simulation as can be seen in section V.

C. Additional BIGhybrid Modules

In BIGhybrid, the *Global Dispatcher* is manual or automatic. In the manual version, the user defines a function for data distributions and a job configuration for each system for both Cloud-BlobSeer and BitDew-MapReduce, such as, input data, data size, chunk size and so on. In automatic release, one *Orchestrator* manages user queries and distributes tasks to systems. A *Global Storage* maintain data users, so the Orchestrator could be initialize a new job, if it is necessary.

The results of the *Global Aggregator* module are implemented as a single Reduce task after the last current Reduce task has been completed. The processing results are tracked and saved in a file for future analysis.

A toolkit for system execution analysis was implemented to assist on creating platforms homogeneous and heterogeneous, and make execution traces based on visualization traces supported from SimGrid. This toolkit enable to users analyze all system execution and change the strategies as needed. The traces can be both individual systems as for all simulation.

V. EVALUATION

This section describes the environment setup and results of evaluation in order to demonstrate the features and scalability of simulator.

A. Environment Setup

Two environments have been considered. The first, in a small scale servers is a proof of concept. The second considers a cluster of 2,000 nodes and evaluates the simulator ability to replicate results obtained in previous work. The proof-of-concept first simulated a homogeneous 5-node cluster with 2 cores each, 5.54 GFlops of processing capacity and 1 Gbps network. This cluster was used to process 2GB of data, 36 maps and 5 reduces. The second cluster contains 5 heterogeneous machines with 2 CPU cores each, which process capacity is drawn from a log-normal distribution according to [20] from 4.76 GFlops to 6.89 GFlops, network of 10 Mbps. This cluster is used to process 1.1 GB of data, 36 maps and 30 reduces.

To define the large scale setup, we used characterization of MR applications performed by Chen [21]. Chen examined MR traces from two production environments from Yahoo and Facebook.

The traces obtained from Yahoo comes from a 2,000 node cluster and contains 30,000 jobs spanning over 3 weeks. The cluster was used to run applications that require bash, interactive and semi-streaming computations. For present work, we model the Yahoo cluster and consider the "aggregate, fast job" applications characterized by Chen. Table II shows the details of these applications, including number of jobs, input data size, job duration, *Map* time and *Reduce* time.

B. Results and Analysis

The first experiment is a proof-of-concept introduced by Figures 4, 5 and 6. Figure 4 shows an execution of Cloud-BlobSeer on the homogeneous cluster. The *Map* tasks produce intermediary keys that are sent during the *shuffle* phase, and *Reduce* tasks initialize once the *Map* tasks are finished.

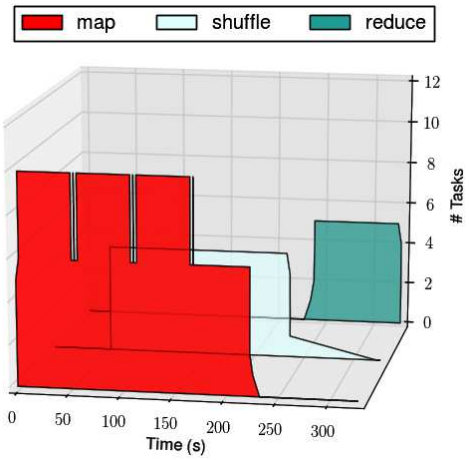

Fig. 4. Job MR on Cloud-BlobSeer simulation

The results of BitDew-MapReduce execution on heterogeneous cluster are show in Figure 5. The *Reduce* tasks start as soon as machines have data to process. It is possible to view that, as the link is 10 Mbps, data transfers take longer to complete during the *shuffle* phase. Execution time is similar to the first cluster because the data *chunk* size on BitDew-MapReduce is 32 MB instead of 64 MB as Cloud-BlobSeer.

Figure 6 shows the execution of BIGhybrid taking into account the two clusters. This figure is specially interesting as it demonstrates the task execution parallelism and the beginning of intermediate data transfers.

Figure 7 shows a job execution of BIGhybrid, for a homogeneous execution, following the specifications of the 2,000 node cluster and application is detailed in Table II. The runtime is 305.13 s to *Map* and 673.32 s to *Reduce*, the and simulation error is $\approxeq$ 5% related to one job in Table II. This demonstrates the scalability of the simulator, reproducibility capacity and its suitability to research of new strategies on hybrid infrastructure.

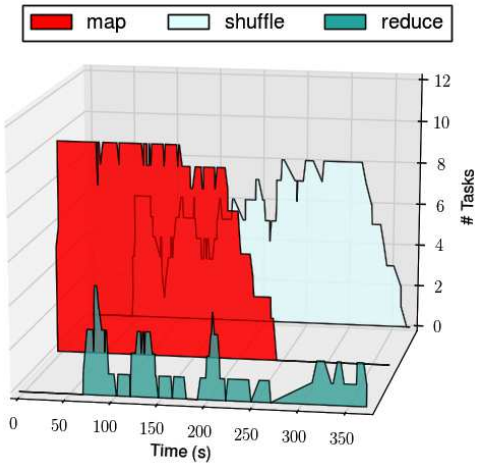

Fig. 5. Job MR on BitDew-MapReduce simulation

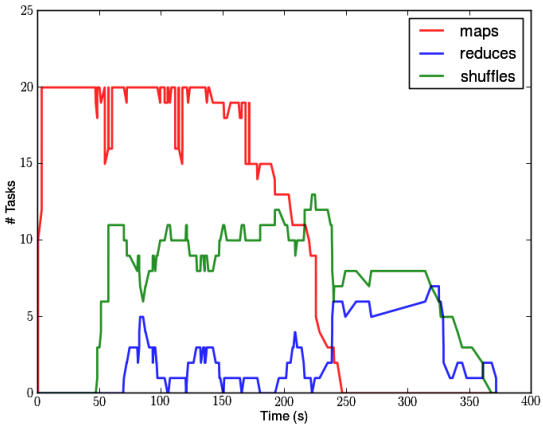

Fig. 6. Job MR on BIGhybrid

VI. Conclusion

The rapid increase in the amount of data produced and current structures will stretch current infrastructure to its limits. Merging both Cloud and DG in hybrid infrastructures can be a feasible low-cost alternative to purely Cloud environments.

BIGhybrid simulator enables the study of strategies for MR in hybrid infrastructures. Initial experiments demonstrate good performance of the simulator and indicate that initial goals have been achieved. However, it is necessary to conduct more experiments to compare the configuration strategies with results from real systems.

The next steps include improving volatility, fault tolerance mechanisms and disk simulation. We also intend to implement virtualization support.

Acknowledgment

This work was partly supported by CAPES Process BEX 14966/13-1. The experiments discussed in this paper were conducted with the aid of the Grid'5000 experimental testbed, under the INRIA ALADDIN development plan with support from CNRS, RENATER and several universities (see https://www.grid5000.fr).

TABLE II. YAHOO TRACES (2,000 MACHINES CLUSTER)

# Jobs	Input	*Shuffle*	Output	Duration	*Map* Time	*Reduce* Time	Label
21,981	174 MB	73 MB	6 MB	1 min	412	740	**Small jobs**
838	568 GB	76 GB	3.9 GB	35 min	270,376	589,385	**Agregate, fast job**
91	206 GB	1.5 TB	133 MB	40 min	983,998	1,425,941	**Expand and aggregate jobs**
35	4.9 TB	78 GB	775 MB	3 hs 45 min	4,481,926	1,663,358	**Data summary**
5	31 TB	937 GB	475 MB	8 hs 35 min	33,606,055	31,884,004	**Data summary, large**
1,330	36 GB	15 GB	4 GB	1 hr	15,021	13,614	**Data transformation**

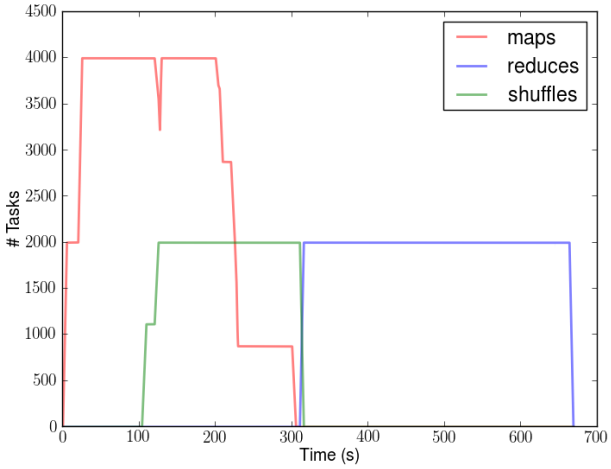

Fig. 7. Job MR on Bighydrid simulation with 2000 hosts

REFERENCES

[1] J. Dean and S. Ghemawat, "MapReduce - A Flexible Data Processing Tool," *Communications of the ACM*, vol. 53, no. 1, pp. 72–77, 2010.

[2] T. White, *Hadoop - The Definitive Guide*, 3rd ed. OReilly Media, Inc., 2012, vol. 1.

[3] P. Mell and T. Grance, "The NIST Definition of Cloud Computing," Tech. Rep., Sep. 2011. [Online]. Available: http://csrc.nist.gov/publications/nistpubs/800-145/SP800-145.pdf

[4] C. Cérin and G. Fedak, Eds., *Desktop Grid Computing*, 1st ed., ser. Numerical Analysis and Scientific Computing. CRC Press, May 2012.

[5] H. Casanova, A. Giersch, A. Legrand, M. Quinson, and F. Suter, "Versatile, Scalable, and Accurate Simulation of Distributed Applications and Platforms," *Journal of Parallel and Distributed Computing*, vol. 74, no. 10, pp. 2899–2917, 2014.

[6] G. Antoniu, J. Bigot, C. Blanchet, L. Bouge, F. Briant, F. Cappello, A. Costan, F. Desprez, G. Fedak, S. Gault, K. Keahey, B. Nicolae, C. Perez, A. Simonet, F. Suter, B. Tang, and R. Terreux, "Scalable Data Management for Map-Reduce-based Data-Intensive Applications: A View for Cloud and Hybrid Infrastructures," *Int. Journal of Cloud Computing*, vol. 2, pp. 150–170, Feb. 2013.

[7] G. Fedak, H. He, and F. Cappello, "BitDew: a programmable environment for large-scale data management and distribution," in *Proceedings of the 2008 ACM/IEEE conference on Supercomputing*, ser. SC '08. Piscataway, NJ, USA: IEEE Press, 2008, pp. 45:1–45:12.

[8] M. Moca, G. Silaghi, and G. Fedak, "Distributed Results Checking for MapReduce in Volunteer Computing," in *Parallel and Distributed Processing Workshops and Phd Forum (IPDPSW), 2011 IEEE Int. Symposium on*, May 2011, pp. 1847–1854.

[9] B. Nicolae, D. Moise, G. Antoniu, L. Bouge, and M. Dorier, "BlobSeer: Bringing high throughput under heavy concurrency to Hadoop Map-Reduce applications," in *Parallel Distributed Processing (IPDPS), 2010 IEEE International Symposium on*, April 2010, pp. 1–11.

[10] W. Kolberg, P. D. B. Marcos, J. C. S. Anjos, A. K. S. Miyazaki, C. R. Geyer, and L. B. Arantes, "MRSG - A MapReduce simulator over SimGrid," *Parallel Comput.*, vol. 39, no. 4-5, pp. 233–244, Apr. 2013.

[11] J. C. S. Anjos, W. Kolber, C. R. Geyer, and L. B. Arantes, "Addressing Data-Intensive Computing Problems with the Use of MapReduce on Heterogeneous Environments As Desktop Grid on Slow Links," in *Proceedings of the 2012 13th Symposium on Computing Systems*, ser. WSCAD-SSC '12. Washington, DC, USA: IEEE Computer Society, 2012, pp. 148–155.

[12] D. Kondo, B. Javadi, A. Iosup, and D. Epema, "The Failure Trace Archive: Enabling Comparative Analysis of Failures in Diverse Distributed Systems," in *(CCGrid), 10th IEEE/ACM Int. Conference on Cluster, Cloud and Grid Computing*. IEEE Computer Society, May 2010, pp. 398–407.

[13] L. Lu, H. Jin, X. Shi, and G. Fedak, "Assessing MapReduce for Internet Computing: A Comparison of Hadoop and BitDew-MapReduce," in *Proceedings of the 2012 ACM/IEEE 13th Int. Conference on Grid Computing*, ser. GRID '12. Washington, DC, USA: IEEE Computer Society, 2012, pp. 76–84.

[14] B. Tang and G. Fedak, "Analysis of Data Reliability Tradeoffs in Hybrid Distributed Storage Systems," in *Parallel and Distributed Processing Symposium Workshops PhD Forum (IPDPSW), 2012 IEEE 26th International*, May 2012, pp. 1546–1555.

[15] J. Ekanayake, S. Pallickara, and G. Fox, "MapReduce for Data Intensive Scientific Analyses," in *IEEE Fourth Int. Conference on eScience*, ser. eScience '08, December 2008, pp. 277 – 284.

[16] C. Dobre and F. Xhafa, "Parallel Programming Paradigms and Frameworks in Big Data Era," *Int. Journal of Parallel Programming*, vol. 42, no. 5, pp. 710–738, 2014.

[17] R. Tudoran, A. Costan, and G. Antoniu, "MapIterativeReduce: A Framework for Reduction-intensive Data Processing on Azure Clouds," in *Proceedings of 3rd Int. Workshop on MapReduce and Its Applications Date*, ser. MapReduce '12. New York, NY, USA: ACM, 2012, pp. 9–16.

[18] T. Hirofuchi and A. Lèbre, "Adding Virtual Machine Abstractions Into SimGrid: A First Step Toward the Simulation of Infrastructure-as-a-Service Concerns," in *Cloud and Green Computing (CGC), 2013 Third International Conference on*, Sept 2013, pp. 175–180.

[19] T. Hirofuchi, A. Lèbre, and L. Pouilloux, "Adding a Live Migration Model into SimGrid: One More Step Toward the Simulation of Infrastructure-as-a-Service Concerns," in *Cloud Computing Technology and Science (CloudCom), 2013 IEEE 5th International Conference on*, vol. 1. IEEE Computer Society, Dec 2013, pp. 96–103.

[20] B. Javadi, D. Kondo, J.-M. Vincent, and D. P. Anderson, "Discovering Statistical Models of Availability in Large Distributed Systems: An Empirical Study of SETI@home," *IEEE Transactions on Parallel and Distributed Systems*, vol. 99, no. PrePrints, 2011.

[21] Y. Chen, A. Ganapathi, R. Griffith, and R. Katz, "The Case for Evaluating MapReduce Performance Using Workload Suites," in *IEEE 19th Int. Symposium on Modeling, Analysis Simulation of Computer and Telecommunication Systems*, ser. (MASCOTS). IEEE Computer Society, July 2011, pp. 390–399.